Aplicación Web de gestión de gráficas y precios de criptomonedas

“Cryptoverse”

TRABAJO FINAL DE GRADO

Autor/a: Adam Power Gonzalez

Tutor/a: Antonio José Reinoso Peinado

Fecha: 2022

ÍNDICE

1.Abstract..... 4
2.Resumen 5
3.Objetivos y motivacion..... 5
4.Antecedentes.....6
4.1. Tabla comparativa 7
5. Análisis de requisitos..... 8
5.1 Tecnologías disponibles 8
6. Propuesta de solución y diseño 15
6.1. Propuesta de solución 15
6.2. Diseño solución propuesta 16
6.3 Pagina principal..... 16
6.4. Página secundaria..... 18
6.5. Diagrama..... 21
7. Plan de trabajo 22
8. Desarrollo de la solución.....23
8.1. API pensada para el proyecto..... 23
8.2. Creación del proyecto..... 24
8.3 Primeras líneas de código..... 24
8.4. Desarrollo de componentes 26
9.. Problemas encontrados 35
10. Evolución y trabajo futuro 35
11. Bibliografía..... 36

1. ABSTRACT

The Purpose of this TFG is the creation of a web page using the Architecture of Redux and the React library and that is in turn to obtain information through an API.
With the trend and direction that cryptocurrencies take these years there are more and more programmers who decide to immerse themselves in this ecosystem so I have proposed to give more visibility to the matter and facilitate the availability of information.
The application must be able to collect information about different cryptocurrencies, display graphs and get the links of these same. I have decided to use the new React and redux library since it is an emerging technology to make a usable and maintainable design over time and it is the one I have been using during my internship and thus put into practice what I have learned. I have proposed to create a web version and another adapted for different mobile devices so that it can be adapted to all types of users comfortably through a more intuitive and visual interface thus achieving a more efficient application. It is intended to give a personal assessment about this architecture and why it is gaining popularity in the software industry.

2. RESUMEN

El Propósito de este TFG es la creación de una página web utilizando la arquitectura de Redux y la librería de React y que está a su vez consiga información a través de una API.
Con la tendencia y el rumbo que llevan las criptomonedas estos años cada vez hay más programadores que deciden sumergirse en este ecosistema asique me he propuesto darle más visibilidad al asunto y facilitar la disposición de la información.
La aplicación debe ser capaz de conseguir recopilar información sobre distintas criptomonedas, mostrar gráficas y conseguir los links de estas mismas.

He decidido usar la nueva librería de React y redux ya que es una tecnología emergente para realizar un diseño usable y mantenible en el tiempo y es la que he estado utilizando durante mis prácticas y así poner en práctica lo aprendido. Me he propuesto crear una versión web y otra adaptada para diferentes dispositivos móviles para que se pueda adaptar a todo tipo de usuarios cómodamente mediante una interfaz más intuitiva y visual consiguiendo así una aplicación más eficiente.
Se pretende dar una valoración personal sobre esta arquitectura y porqué está ganando popularidad en la industria del software.

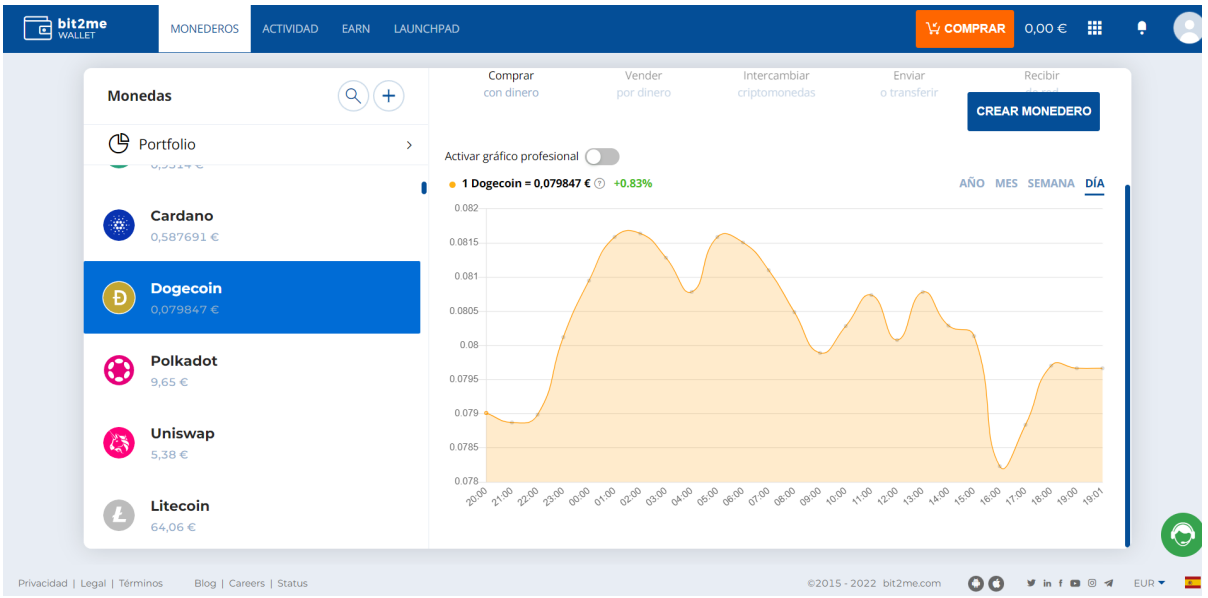
3. Objetivos y motivación

Como ya he comentado, el ecosistema de las criptomonedas está en pleno auge, en constante evolución y por lo tanto en un continuo cambio. A mí, personalmente me fascina este ecosistema y agradezco cuando una web, aplicación o foro funciona correctamente, o tiene una tecnología innovadora detrás de ello.
Es por esto que me propuse sacar adelante mi propia página, donde se pudiese obtener todo tipo de información sobre criptomonedas, su valor actualizado y sus diferentes links.
Como reto personal, quise utilizar React, librería que tuve que aprender desde cero, de manera que me obligué a mi mismo a descubrir todo lo involucrado con este lenguaje, desde sus distintas librerías hasta todas sus funciones.
Es un objetivo que creo haber cumplido.

4. Antecedentes

A la hora de buscar el detalle que marcase la diferencia entre mi página web y las convencionales que he ido encontrando, como Bit2me: https://bit2me.com/ (empresa española que tomaré como ejemplo) , me di cuenta de que no mostraban cierta información que para mi era necesaria.

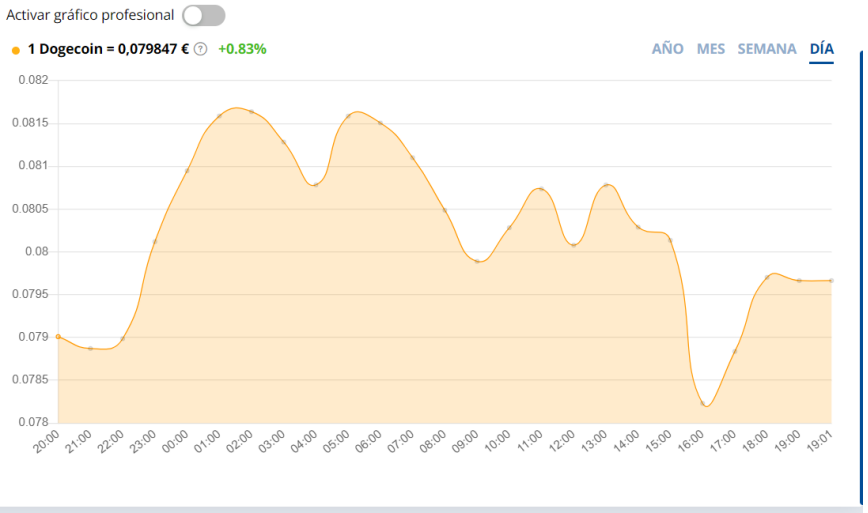
Un ejemplo de esto puede ser el tipo de temporalidades que tenían en la gráfica o que no mostraran el volumen que había en 24 horas.



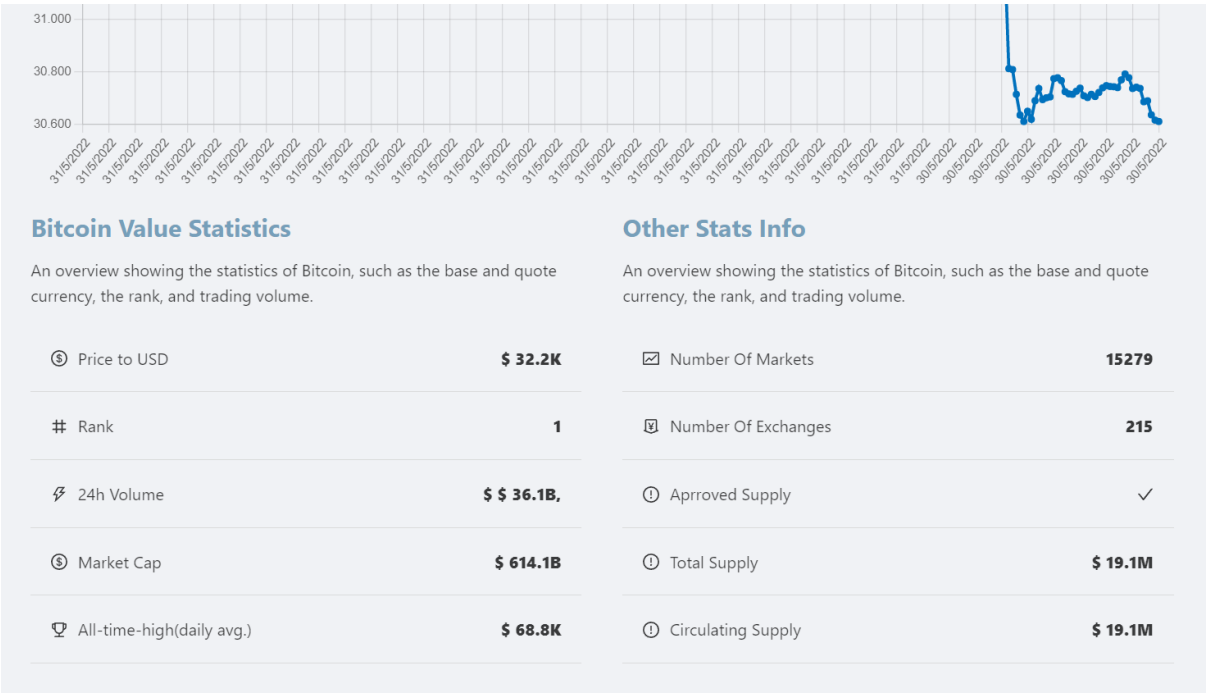
(Vista general de Bit2me)

Solo muestran una gráfica con el precio, el nombre de la criptomoneda y escasas temporalidades, así que decidí que este era el ejemplo perfecto para comparar.

A continuación muestro en la siguiente captura que no muestran ningún detalle o información adicional en la parte inferior de la gráfica.



En el caso de mi página web contemplo la posibilidad de añadir más información detallada para cada criptomoneda a través de una API:



Como se puede observar en la captura, aparece información mucho más detallada y actualizada.

4.1. Tabla comparativa:

Características	Bit2me	Cryptoverse
Temporalidad	Sí	Sí
Precio	Sí	Sí
Ranking	No	Sí
Volumen 24 horas	No	Sí
Capitalización del mercado	No	Sí
Máximo histórico	No	Sí
Nº de exchanges	No	Sí
Abastecimiento total	No	Sí
Total en circulación	No	Sí
Links redes sociales	No	Sí
Historia de la moneda	No	Si

5. Análisis y especificación de requisitos

En este capítulo de la memoria describiremos las características que debe tener la aplicación, también los objetivos que debe cumplir.
En primer lugar, debe ser una aplicación a tiempo real, ya que se necesita que todos los usuarios involucrados en el uso del programa dispongan de la misma información en todo momento.
Debe ser capaz de mostrar todas las criptomonedas con sus diferentes precios a tiempo real pudiendo configurar en cada una de ellas distintos valores como la temporalidad diaria o el número que se muestran.
Por último, debe tener una interfaz gráfica intuitiva y sencilla para garantizar un uso eficiente y comprensión de la misma.

5.1 Tecnologías disponibles:

5.2 React: React es una librería Javascript focalizada en el desarrollo de interfaces de usuario. Así se define la propia librería y evidentemente, esa es su principal área de trabajo. Sin embargo, lo cierto es que en React encontramos un excelente aliado para hacer todo tipo de aplicaciones web,o incluso aplicaciones para móviles. Para ello, alrededor de React existe un completo ecosistema de módulos, herramientas y componentes capaces de ayudar al desarrollador a cubrir objetivos avanzados con relativamente poco esfuerzo.
Por tanto, React representa una base sólida sobre la cual se puede construir casi cualquier cosa con Javascript. Además facilita mucho el desarrollo, ya que nos ofrece muchas cosas ya listas, en las que no necesitamos invertir tiempo de trabajo. En este artículo te ampliaremos esta información, aportando además diversos motivos por los que usar React como librería del lado del cliente.React es una librería desarrollada inicialmente por Facebook. Es software libre y a partir de su liberación acapara una creciente comunidad de desarrolladores y entusiastas. Su creación se realizó en base a unas necesidades concretas, derivadas del desarrollo de la web de la popular red social.

Además de facilitar el desarrollo ágil de componentes de interfaces de usuario, el requisito principal con el que nació React era ofrecer un elevado rendimiento, mayor que otras alternativas existentes en el mercado. Detectaron que el típico marco de binding y doble binding ralentizaba un poco su aplicación, debido a la cantidad de conexiones entre las vistas y los datos. Como respuesta crearon una nueva dinámica de funcionamiento, en la que optimizaron la forma como las vistas se renderizan frente al cambio en los datos de la aplicación.

A partir de ahí la probaron en su red social con resultados positivos y luego en Instagram, también propiedad de Facebook. Más adelante, después de su liberación y alentados por los positivos resultados en el rendimiento de React, muchas otras aplicaciones web de primer nivel la fueron adoptando.

Uno de los objetivos de React es que Sirve para desarrollar aplicaciones web de una manera más ordenada y con menos código que si usas Javascript puro o librerías como jQuery centradas en la manipulación del DOM. Permite que las vistas se asocien con los datos, de modo que si cambian los datos, también cambian las vistas.

5.3 Instalación de react: A continuación procedere con la instalación de react, utilizando :

```
npx create-react-app nombreDeApp
```

-npx create-react-app

npx: Es una herramienta de ejecución de paquetes de npm, nos permite instalar grandes paquetes sin necesidad de instalarlos de forma global en nuestro equipo.

create-react-app: es un paquete creado por el equipo de react que viene con todas las dependencias necesarias y una estructura básica para ejecutar un proyecto en react.

En “nombre de la app” es donde escribiremos nuestro nombre de la aplicación pero eso se detalla más adelante.

5.4 Redux: Redux es un patrón de arquitectura de datos que permite manejar el estado de la aplicación de una manera predecible. Está pensado para reducir el número de relaciones entre componentes de la aplicación y mantener un flujo de datos sencillo.

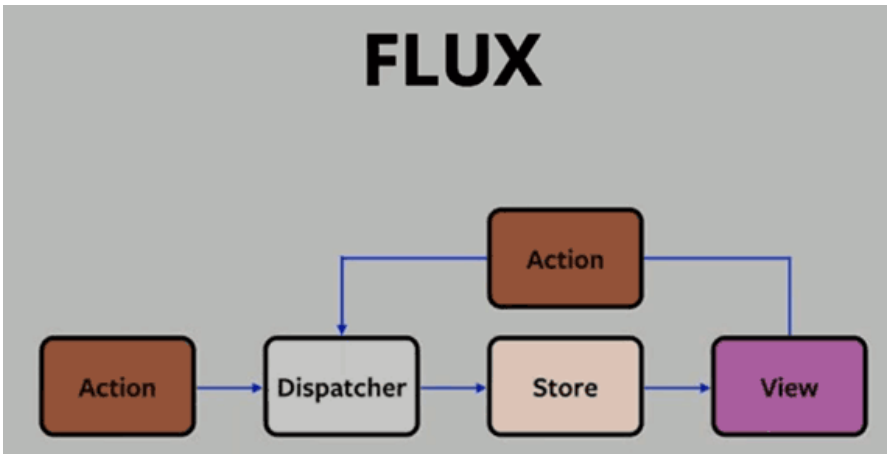
Iniciado originalmente por la comunidad de React, como evolución y mejora de las ideas de Flux, Redux se ha convertido en un patrón transversal, capaz de adaptarse a cualquier tipo de librería o framework del lado del cliente. Incluso se podría usar sin necesidad de otro framework Javascript. También se puede ejecutar del lado del servidor o en aplicaciones para móviles, por lo que sus ámbitos de aplicación son muy amplios.

Redux es una librería ligera, de apenas 2 KB de peso, por lo que nos resulta ligera también para su ejecución por el motor de Javascript. Su comunidad es muy amplia y ha sido adoptado con éxito por grandes empresas como Netflix.

Estos son los beneficios que aporta la aplicación del patrón de Redux.

- Arquitectura escalable de datos
- Mayor control sobre el flujo de datos y el estado de la aplicación
- Estado global e inmutable

Facebook, con la intención de simplificar los modelos actuales y hacer más fácil de predecir el flujo de los datos en la aplicación, lanzó Flux. Flux tiene su característica más destacable en el flujo de los datos, ya que éste se realiza siempre en una única dirección.

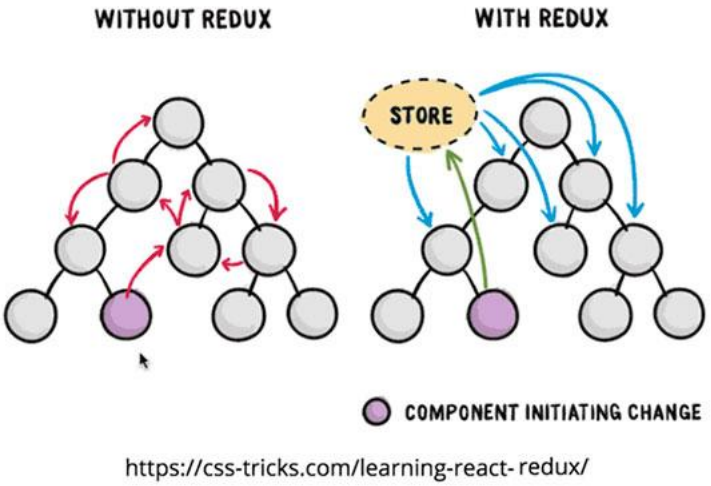


En resumen, este diagrama nos dice que el "Store" contiene todos los datos de la aplicación, su estado. Los datos siempre van hacia las vistas en una única dirección y para modificar los datos desde la vista se lanzan acciones. Esas acciones, una vez tratadas por el dispatcher, pueden producir un cambio de Estado en el Store, que a su vez viajará hacia la vista.

Flux es solamente una declaración de intenciones, un patrón. Redux es una implementación del patrón, en una librería ligera que podemos usar en cualquier tipo de ambiente de desarrollo, framework, etc. El diagrama con Redux evoluciona un poco, pero mantiene la esencia comentada para Flux: el flujo unidireccional de los datos y la necesidad de solicitar cambios en los mismos por medio de acciones.

Para concluir esta introducción puedes ver en el siguiente diagrama. Muestra de una manera muy visual cómo la arquitectura de componentes varía con y sin Redux. Por un lado tenemos el diagrama de componentes y flujo de los datos en una arquitectura tradicional, con data-biding de una y dos direcciones, donde muchos componentes son capaces de manipular el estado.

Al otro lado tienes una arquitectura de componentes basada en Redux, donde hay un store que alimenta a todos los componentes y donde cualquier modificación se realiza por medio de una acción.



(cómo la arquitectura de componentes varía con y sin Redux)

5.5 Paquetes de instalación de redux

Debemos ejecutar los siguientes comandos en un terminal nuevo ingresando en la carpeta que vayamos a instalar mediante -cd cryptoverse.

Una vez dentro debemos ejecutar las siguientes líneas de código:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\adamp\OneDrive\Escritorio\project_cryptoverse-main> npm install react-redux @reduxjs/toolkit
```

(esto instalará el paquete de Redux y nos ayudara a obtener datos de la API)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\adamp\OneDrive\Escritorio\project_cryptoverse-main> npm install axios/chart.js/html-react-parser/millify/moment/react-chartjs-2
```

(Por último estos seran las librerías necesarias que explicare a continuación)

-**Npm install axios**: Axios se basa en promesas, lo que le permite aprovechar async y await de JavaScript para obtener un código asíncrono más legible.

También puede interceptar y cancelar solicitudes, y hay una protección integrada del lado del cliente contra la falsificación de solicitudes entre sitios.

-**Npm install chart.js** :es una librería para crear gráficas dinámicas

-Npm install html-react-parser: convierte una cadena HTML en uno o más elementos React.

-Npm install millify: es un paquete que va a transformar números grandes en cadenas legibles

-**Npm install moment:** es una increíble biblioteca de JavaScript que te ayuda a administrar las fechas, tanto en el navegador como en Node.js.

-**Npm install react-chartjs-2 :** Elegimos Chart JS2 como la biblioteca para los gráficos

5.6 Node.js: Node.js es un entorno de tiempo de ejecución de JavaScript, de ahí su terminación «.js». Este entorno de tiempo es open source, es decir, de código abierto, multiplataforma y que se ejecuta del lado del servidor.

Este entorno fue creado por los desarrolladores de JavaScript con el objetivo de ir un paso más allá con este lenguaje de programación.

Hasta la creación de Node.js, allá por el año 2009, el lenguaje de programación JavaScript únicamente podía ejecutarse del lado del navegador o cliente,

Como JavaScript únicamente se podía utilizar dentro del marco de las etiquetas <script> </script>, los desarrolladores tenían que tirar de diferentes lenguajes y herramientas tanto en el frontend como en el backend.

Node.js posee todo lo necesario para ejecutar código JavaScript del lado del servidor. Algo que facilita mucho el trabajo de los desarrolladores y el motivo por el que actualmente sea una de las herramientas de trabajo más usadas en desarrollo web.

Node.js sirve para crear sitios web dinámicos muy eficientes, escritos con el lenguaje de programación JavaScript. Normalmente, los desarrolladores se decantan por este entorno de ejecución cuando buscan que los procesos se ejecuten de forma ágil y sin ningún tipo de bloqueo cuando las conexiones se multiplican.

Por ejemplo, en aplicaciones IOT (Internet de las cosas), aplicaciones de transmisión de datos, aplicaciones basadas en REST API o aplicaciones de mensajería instantánea lo más utilizado es Node.js, ya que resulta muy estable y rápido frente a miles de conexiones simultáneas.

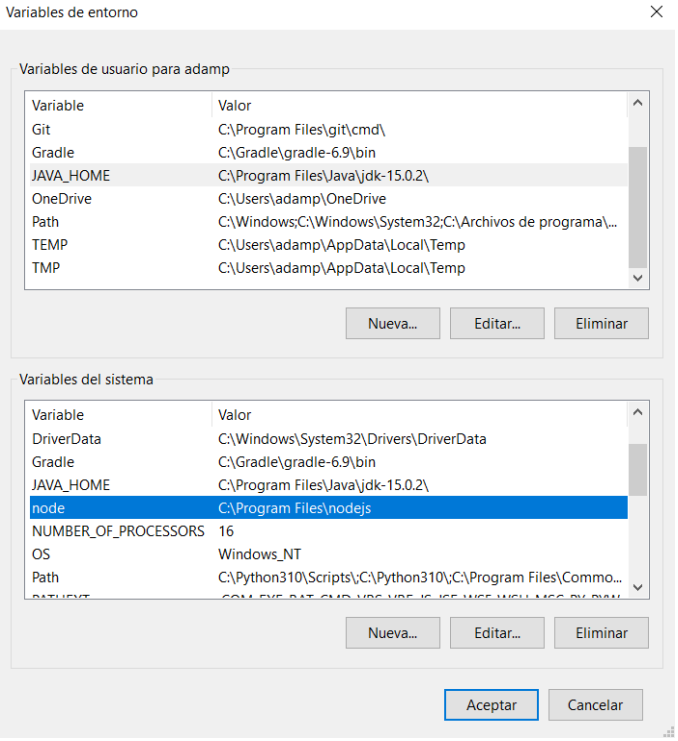
5.7 Instalación de Node.js

1. Entrar en https://nodejs.org/es/download/ y descargar el instalador de Node.js

2. Ejecutar el instalador que acabamos de descargar. Simplemente debemos avanzar en el proceso de instalación.

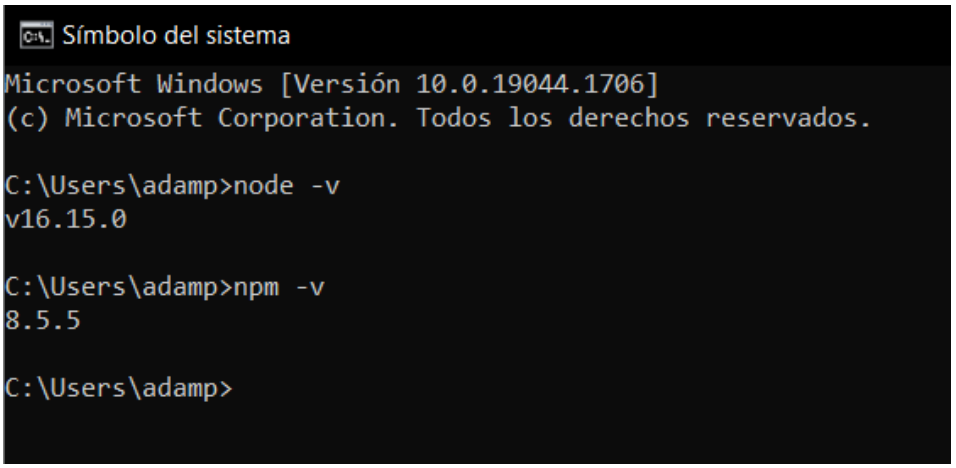
3.Y el último componente es agregar Node a la variable PATH de Windows, este paso puede ser un poco lioso, hay que copiar la ruta donde se ha instalado Node y copiarla en el Path que se encuentra en:
Inicio / Configuración / Acerca de / configuración avanzada de sistema/ Variables de entorno

(Como se muestra en la siguiente imagen)



4.Una vez finalizado el proceso de instalación, podemos comprobar fácilmente si se nos ha instalado correctamente. Para ello, vamos al intérprete de comandos de nuestro ordenador (en Windows, por ejemplo, escribir “cmd” en la barra de búsqueda y abrir la aplicación de “Símbolo del sistema”).

5.En la ventana de comandos, escribir node -v y pulsar la tecla Enter. Nos debería aparecer la versión que tenemos instalada de Node.js. Para comprobar que se nos ha instalado también NPM, escribiremos npm -v y pulsaremos de nuevo Enter. Nos debería aparecer también en este caso la versión del Node Package Manager.



(Como se muestra en la imagen)

6. Propuesta de solución y diseño

6.1 Propuesta de solución

Para llevar a cabo todo lo dicho anteriormente, llevo a cabo una planificación de cómo va a ser mi página web.

Esta tiene que ser lo suficientemente sencilla e intuitiva para que se adapte a todo tipo de usuarios .

A continuación explico de qué elementos va a contar la página web para llevar a cabo estos objetivos.

Como ya podrás imaginar voy a utilizar React con Redux y un poco de CSS para darle una interfaz más visual.

6.2 Diseño solución propuesta

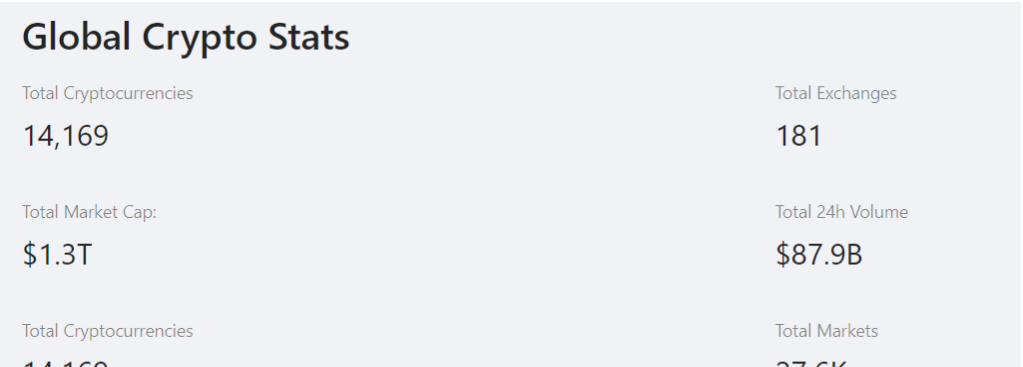
6.3 Página principal

La página para comenzar tiene que incluir un menú y un logotipo en la parte izquierda de la pantalla.
Para este menú irán incluidos dos elementos que uno será un link a nuestra página principal, y el siguiente será un enlace a todas las criptomonedas disponibles, también tendrá que tener un título.

Como mostraré en la siguiente imagen.

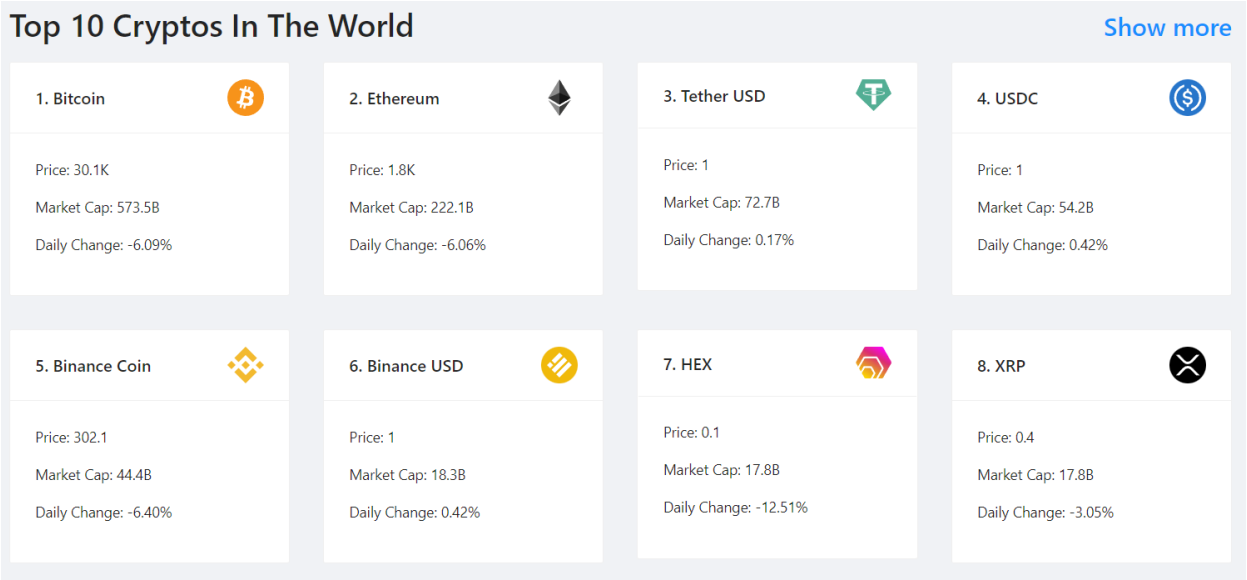
Lo siguiente que incluirá la página web es una breve descripción con información de lo que se va a ver a continuación.
Detalles de interés general como cuántas criptomonedas hay, que capitalización bursátil hay en total o cual ha sido el volumen de las últimas 24 Horas. Esto irá en la parte superior de la página próximo al menú.

como mostraré en la siguiente imagen



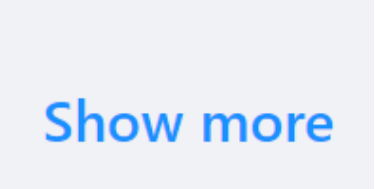
A continuación en la parte inferior tendremos información y conocimiento sobre cual es el top 10 de criptomonedas con mayor capitalización bursátil, estas irán variando a tiempo real a través de la información que obtenga de la API.

como mostraré en la siguiente imagen

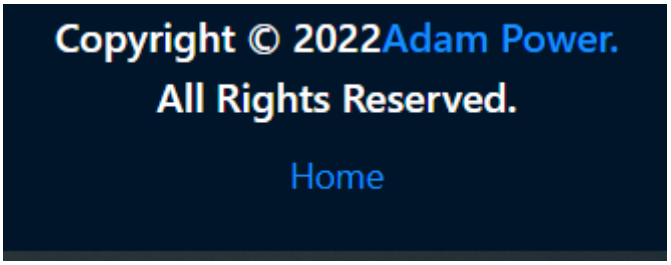


También contaremos con un botón para enseñar todas las criptomonedas en el caso de que queramos hacerlo, en la parte superior derecha.

Este tendrá este aspecto



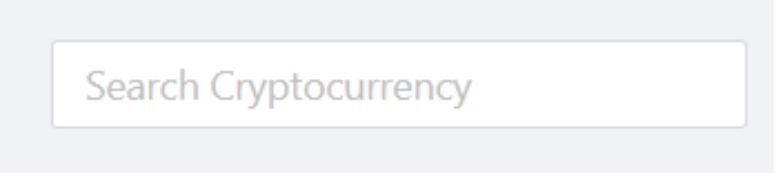
Y por último para acabar con la página principal incluire en la parte inferior de la pagina web creare un footer con distintos datos y un link para poder ir a nuestra página principal.



6.4 Página secundaria

Mi segunda pagina se llamara criptomonedas, permanecerá con el mismo menú ya que este será inamovible, la idea es hacerlo lo más sencillo posible.
Por lo tanto el diseño de la segunda página será parecido pero con más información proporcionada.
La parte superior tendrá un panel de búsqueda que servirá para buscar cualquier criptomoneda que se nos ocurra.

como muestro en la siguiente imagen

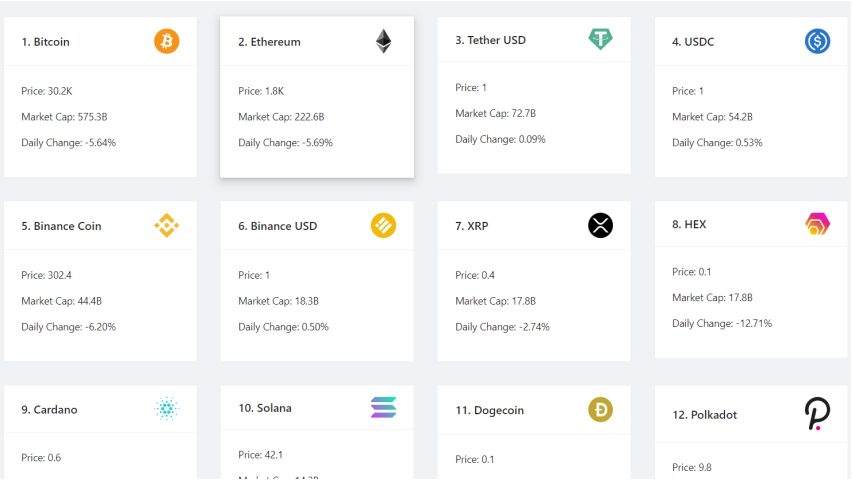


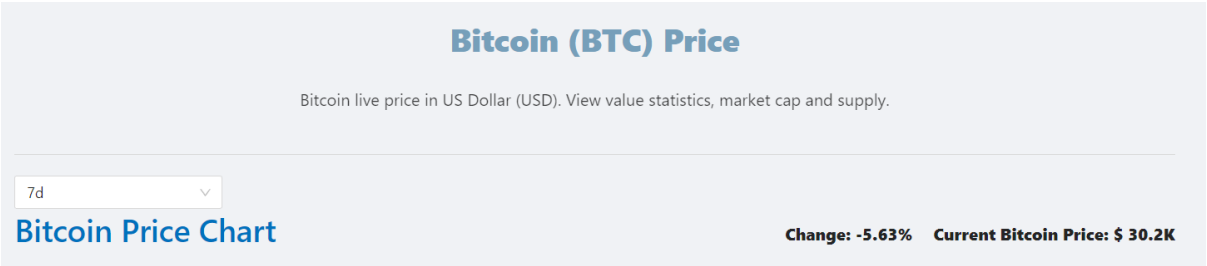
A continuación deberá mostrarse las criptomonedas disponibles en diferentes recuadros y ordenadas según su capitalización de mercado, en ese recuadro se mostrará una información adicional como cuánto ha variado en un día o el precio que tienen.

como muestro en la siguiente imagen

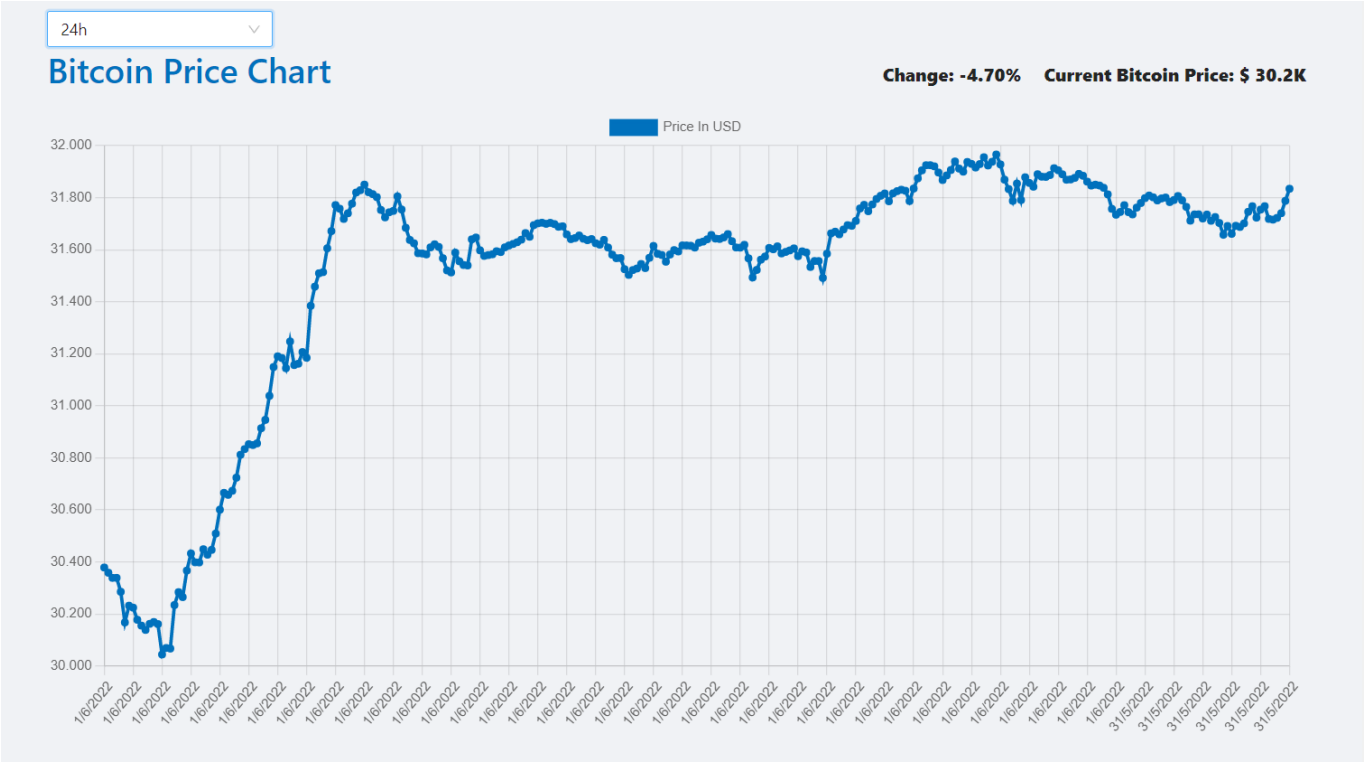
Lo siguiente que quiero enseñar es qué información muestro si pulsas en cualquiera de ellas. Para ello la primera cosa que se debería enseñar en la parte superior es un breve descripción de qué criptomoneda es la que estoy visualizando.

En la siguiente imagen se muestra la parte superior de la pantalla



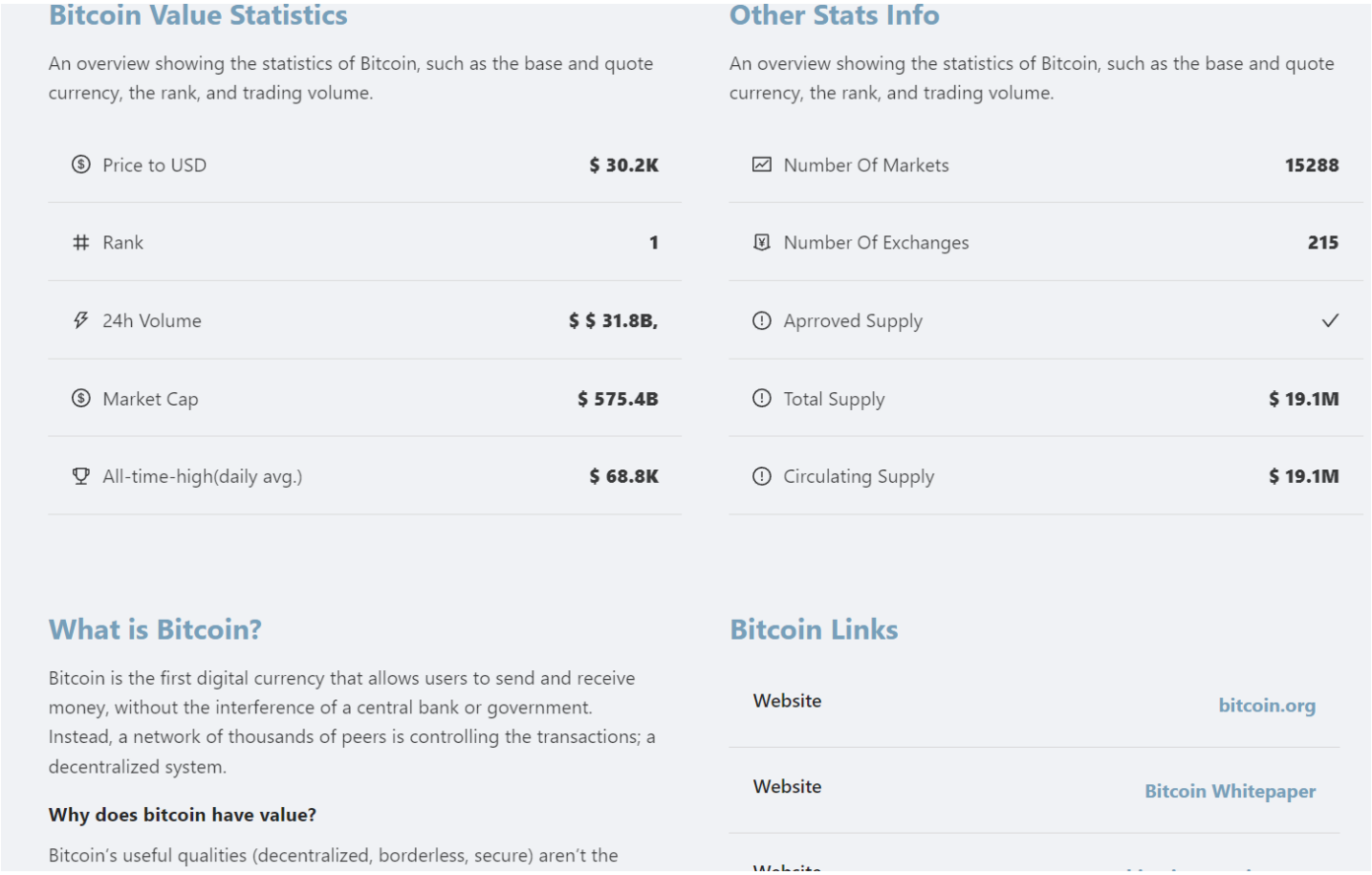


A continuación lo que se deberá mostrar es la gráfica con el historial de precios y una casilla en la que podamos seleccionar diferentes temporalidades para verlo a lo largo del tiempo.
En esta imagen se muestra el historial gráfico con la casilla de seleccionar temporalidades.



El precio vendrá indicado en dólares ya que es la principal divisa que se utiliza en los mercados bursátiles.
Y por último para acabar con mi página tendremos diferentes secciones donde podremos acceder a un sin fin de información como los detalles y la distribución del token.
Otro apartado donde tendremos la tecnología usada para el proyecto y la historia detrás del proyecto y otro apartado donde tendremos links a sus redes sociales, página web y comunidad.

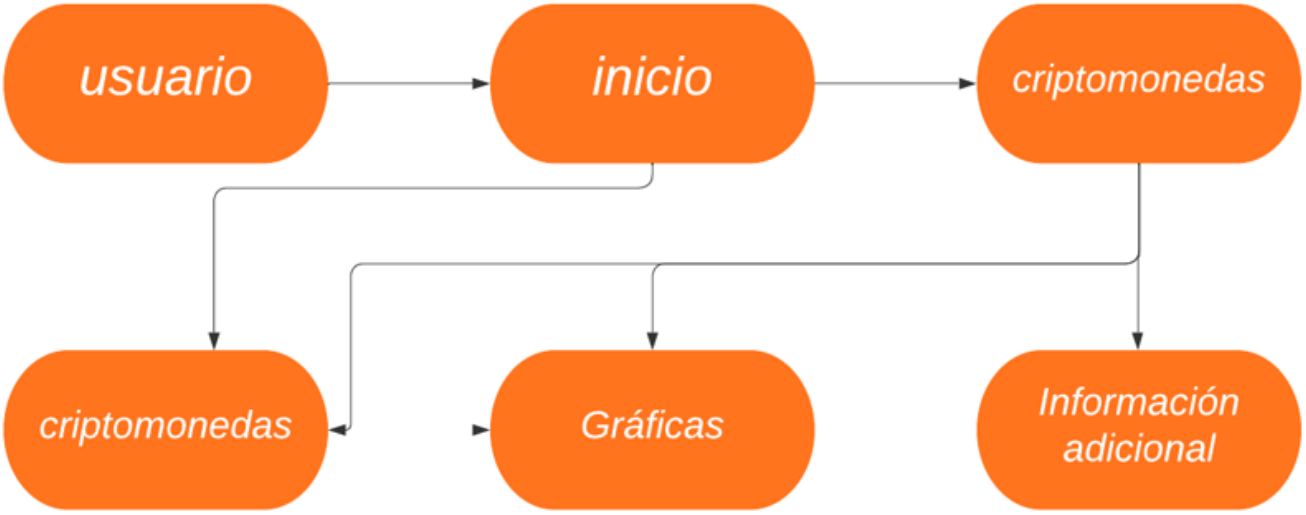
Esta última parte de la página web irá organizada en dos columnas y que cada una de ellas ocupe la mitad de la página como se mostrará en la siguiente imagen.



En la parte inferior también deberá tener el footer para poder volver al principio e incluiré también derechos de autor.

6.5 Diagrama

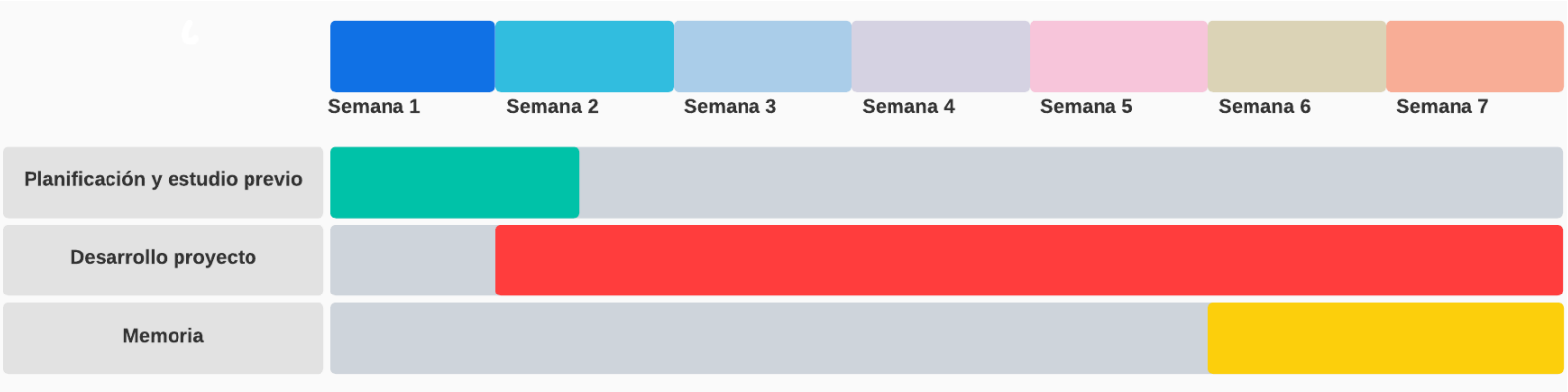
A continuación muestro cómo se relacionaría un usuario con la página web y que podrá encontrar dentro de ella.



En esta se puede observar que si el usuario entra en el inicio , este solo podría ver el top 10 de criptomonedas, mientras que si se mete en la sección de criptomonedas puede acceder a más información como gráficas o información adicional.

7. Plan de trabajo

Decido hacer un diagrama de Gantt para ilustrar el tiempo dedicado a cada parte del proyecto.

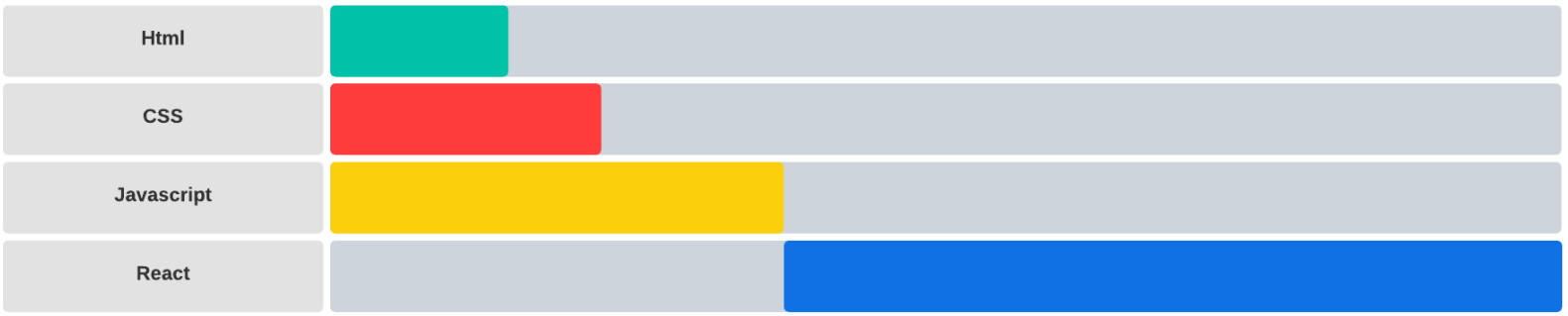


Como se puede observar en el siguiente diagrama estuve durante una semana y media mas o menos ideando que quería hacer, hasta que me di cuenta de que quería hacer una pagina web relacionada con las criptomonedas.

El desarrollo del proyecto empezó casi a la vez que la planificación porque quería comprobar si eran factibles las ideas que estaba teniendo y hasta dónde podía llegar. Para ello me di cuenta de que necesitaba React con alguna librería que me ayudara. Decidí usar react porque vi que es uno de los lenguajes más utilizados en este entorno cripto y para el Web3 (siguiente generación de internet), aun así el desarrollo del proyecto ha ocupado toda la planificación hasta dia de hoy que sigo resolviendo pequeños fallos y seguiré con la evolución y el proceso en un futuro próximo

A la memoria del proyecto decidí dedicar las últimas dos semanas hasta dia de hoy, ya que no quería empezarlo hasta que tuviese todo el desaroollo bien y pudiese mostrar algo de cual estaba siendo el resultado.

Y a continuación presento otro diagrama de Gantt del uso de las tecnologías que han sido utilizadas para llevar a cabo el proyecto.



8.Desarrollo de la solución

8.1. API pensada para el proyecto

Decido utilizar la pagina web de rapidApi (https://rapidapi.com/) , web que he utilizado antes para diferentes proyectos y que siempre recomiendo debido a la cantidad y variedad de APIs que disponen en su portal web.
La API que elijo al final es (https://rapidapi.com/Coinranking/api/coinranking1/)
Coinranking API es una Api que te permite tener disponible un montón de información sobre distintas criptomonedas como el precio de estas mismas , gráficos históricos etc.
El único problema que tuve y que fue a mitad del proyecto es que me habría encantado incluir un noticiario de criptomonedas y algo más de información como distintos intercambios donde poder encontrar la criptomoneda que buscabas.

Pero desafortunadamente si querías conseguir información sobre lo mencionado anteriormente necesitabas realizar una suscripción y pago, lo cual me negué a hacerlo porque sabía que podía conseguir esta información gratuitamente a través de otras APIs.

8.2. Creación del proyecto

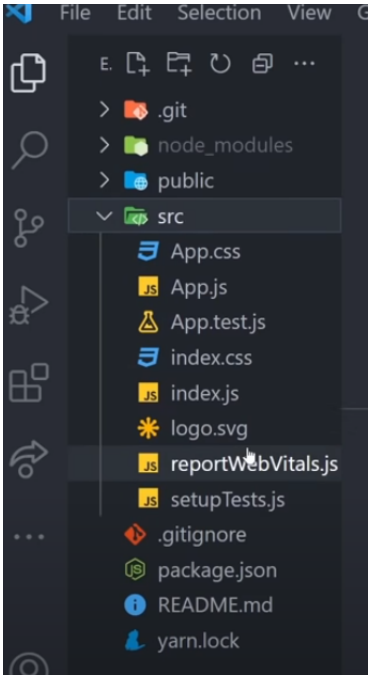
Lo primero que tuve que hacer para empezar a crear mi página web fue crear una carpeta nueva en el escritorio. Aquí es donde va a estar alojado lo que es todo mi proyecto junto con sus diferentes componentes.
Tras un rato pensando llegue a la conclusión de que iba a nombrarlo cryptoverse, quise juntar la palabra Metaverse (Metaverso) con criptomonedas (crypto en Inglés) y surgió este resultado.

8.3. Primeras líneas de código

Para comenzar con cualquier proyecto de React lo primero que hay que hacer es abrir el terminal de nuestro IDE (en mi caso Visual Studio Code) y escribir el siguiente comando.

```
npx create-react-app nombreDeApp
```

Esto nos va a crear una aplicación de react vacía dentro de nuestro directorio.
Una vez creada me debería quedar así.
Una vez tenemos esto nuestra carpeta más importante va a ser la de src, aquí es donde construiré todos los componentes de mi aplicación.
Lo próximo que hago es eliminar todas las carpetas creadas por defecto y empiezo a crear distintos componentes que me vayan a ser necesarios.



El primer componente que creo es el index.js , es el más importante ya que el punto de partida para cualquier aplicación de React.
Empezaré escribiendo las siguientes líneas de código

```
src > index.js
You, seconds ago | 1 author (You)
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';        You, seconds ago • Uncommitte

ReactDOM.render(<App />, document.getElementById('root'));
```

Los primeros import son para cargar la aplicación con react y que seamos capaces de utilizar React. Lo siguiente sera importar la App y por último el ReactDOM.render lo que hacemos es pasar esta aplicación como un componente y luego como un parámetro y con getElementById será la raíz para conectar nuestro rootdiv en html que es donde irá toda nuestra aplicación.
El siguiente paso será cambiar el título en el Html que React nos genera por defecto por cryptoverse.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="./cryptocurrency.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using create-react-app" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Cryptoverse</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <!-- Donde va la aplicacion -->
    <div id="root"></div>
  </body>
</html
```

También cambiar el link de la imagen que vamos a mostrar como logo y en la parte inferior de la imagen es en ese div donde enrutamos toda nuestra aplicación.

Lo siguiente será crear otro archivo en nuestro folder de src, el archivo se llamará App.js, este será un componente funcional básico, aquí es donde enrutamos todos los componentes que vaya creando más adelante, será como la página principal que se muestre.

Dentro de App.js importar las librerías principales que voy a usar de React y Redux y creó distintos componentes como la barra de navegación o el pie de página.

```
import React from 'react';
import { Switch, Route, Link } from 'react-router-dom';
import { Layout, Typography, Space } from 'antd';

const App = () => {
    return (        You, seconds ago • Uncommitted changes
        <div className="app">
            <div className="navbar">

            </div>
            <div className="main">

            </div>
            <div className="footer">

            </div>
        </div>
    );
}
```

8.4. Desarrollo componentes

Lo siguiente que haré será crear los diferentes componentes que va a tener mi aplicación, por lo tanto procedo a crear distintos archivos que vaya a necesitar.
Basado en el diseño y la idea que tengo necesito crear distintos archivos para poder utilizarlos luego.

Necesitare una carpeta para las criptomonedas ya que serán uno de mis elementos que pretendo mostrar.

Tambien tendre que tener otro archivo que se llamara detalles de criptomonedas (Crypto Details), desde donde obtendrás toda la información de las distintas criptomonedas, creare distintas columnas donde se muestre la información indicada anteriormente y creare distintas temporalidades para poder elegir en las gráficas.

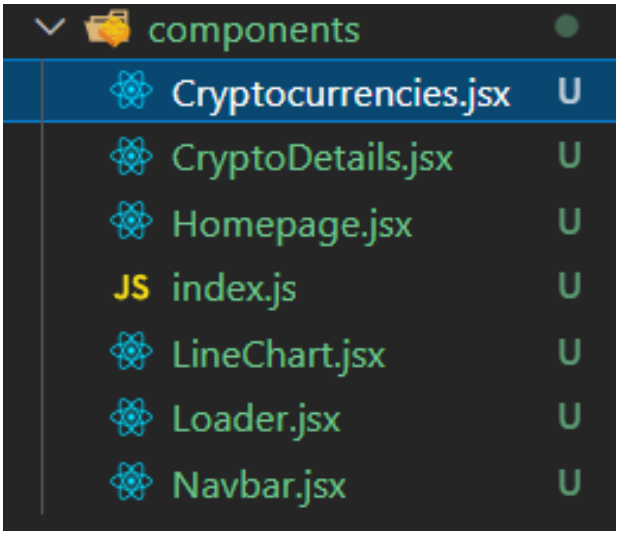
También crearé otro componente archivo que será mi página principal y la página de bienvenida. En ella mostraré el top 10 de criptomonedas.

Creare también otro componente que se llamara gráficas (Linechart) que será donde cree los elementos que quiera mostrar en mi gráfica.

Otro componente que será de diseño, será como nuestro cargador de la página.

Y por último un componente que se llamara barra de navegación (NavBar) que este va a estar compuesto por los elementos que vayan dentro de mi menú, también incluire el logotipo y donde adaptare el diseño de la interfaz en caso de ser usado por dispositivos móviles.

Al final tiene que haber los siguientes componentes como se muestra en la siguiente imagen.



Lo siguiente que haré ser trabajar en mi componente NavBar que mete el logo del menú, el pie de página , y diferentes links para llevarte al inicio.

```
cryptoverse > src > components > navbar.jsx > navbar
1   import React from 'react';
2   import{ Button, Menu, Typography, Avatar } from 'antd';
3   import { Link } from 'react-router-dom';
4   import { HomeOutlined, MoneyCollectOutlined, BulbOutlined, FundOutlined, MenuOutlined } from '@ant-desing/icons';
5
6   import Icon from '../images/cryptocurrency.png';
7   const navbar = () => {
8     return (
9       <div className="nav-container">
10          <div className="logo-container">
11              <Avatar src={icon} size="lage" />
12              <Typography.Title level={2} className="logo">
13                  <Link to="/">Cryptoverse</Link>
14              </Typography.Title>
15
16          </div>
17          <Menu theme="dark">
18          <Menu.Item icon={<homeOutlined />}>
19                  <Link to="/">Home</Link>
20              </Menu.Item>
21              <Menu.Item icon={<FundOutlined />}>
22                  <Link to="/cryptocurrencies">Cryptocurrencies</Link>
23              </Menu.Item>
```

Lo siguiente que tendré que hacer es crear la estructura principal desde App.js.
Tendré que hacer import de todos los componentes que vayan a ir dentro de la App para poder enlazarlos. También necesitare hacer import de las librerías que vaya a necesitar que este caso serán switch, route y link de React-dom para poder enlazarlos de maneras diferentes y dinámicas. La siguiente librería que usaré será Layout, Typhography y Space de ant design que es una librería de diseño de React.

```
App.js > App
                    <div className="routes">
                        <Switch>
                            <Route exact path="/">
                                <Homepage />
                            </Route>
                            <Route exact path="/exchanges">
                                <Exchanges />
                            </Route>
                            <Route exact path="/cryptocurrencies">
                                <Cryptocurrencies />
                            </Route>
                            <Route exact path="/crypto/:coinId">
                                <CryptoDetails />
                            </Route>
                        </Switch>
                    </div>
                </Layout>
            </div>
            <div className="footer">
```

El siguiente paso será crear los componentes de mi página principal. Procedo por hacer la parte superior donde se encontrará la información general de todo el mercado cripto.

```
import React from 'react'
import millify from 'millify';
import { Typography, Row, Col, Statistic } from 'antd';
import { Link } from 'react-router-dom';

const { Title } = Typography;

const Homepage = () => {
  return (
    <>
        <Title level={2} classname="heading">Global Crypto Stats</Title>
        <Row>
            <Col span={12}><Statistic title="Total Cryptocurrencies" value="5"/></Col>
            <Col span={12}><Statistic title="Total Exchanges" value="5"/></Col>
            <Col span={12}><Statistic title="Total Market Cap" value="5"/></Col>
            <Col span={12}><Statistic title="Total 24h Volume" value="5"/></Col>
            <Col span={12}><Statistic title="Total Markets" value="5"/></Col>

        </Row>
    </>
  )
}
```

Utilizo las librerías necesarias pero cuando esté más avanzado necesitaré más. también añadiré diferentes valores que serán de dónde sacarán los datos de la API.

Lo siguiente a lo que procedo es en crear una nueva carpeta que se llame servicios, ya que es la forma que Redux tiene para obtener datos, aquí es donde irán mis archivos con los que enlazar toda la aplicación.

Crearé un archivo llamado CryptoApi que es de donde se obtendrá la lógica de la Api, para ello tengo que copiar el código que la Api me proporciona. También incorporaré mi archivo CSS en esta parte donde iré enlazando todo con diferentes containers, ya que sino en React no se muestra nada.

Procedo a desarrollar la Api que menciono antes, quedaria algo asi.

```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react';
{/* variable don metemos los headers de la Api */}
const cryptoApiHeaders = {
  'x-rapidapi-host': process.env.REACT_APP_CRYPTO_RAPIDAPI_HOST,
  'x-rapidapi-key': process.env.REACT_APP_RAPIDAPI_KEY,
};
const createRequest = (url) => ({ url, headers: cryptoApiHeaders });
{/* Creo opciones dentro de un objeto */}
export const cryptoApi = createApi({
  reducerPath: 'cryptoApi',
  baseQuery: fetchBaseQuery({ baseUrl: process.env.REACT_APP_CRYPTO_API_URL }),
  endpoints: (builder) => ({
    getCryptos: builder.query({
      query: (count) => createRequest(`/coins?limit=${count}`),
    }),

    getCryptoDetails: builder.query({
      query: (coinId) => createRequest(`/coin/${coinId}`),
    }),
 //  obtengo el historial de la criptos desde la API
    getCryptoHistory: builder.query({
      query: ({ coinId, timeperiod }) => createRequest(`coin/${coinId}/history?timeperiod=${timeperiod}`),
    }),
  })
```

Como se puede observar en la siguiente imagen exportar las variables que me proporciona la Api y creo una request de que enlace los headers de la Api. Al final del todo lo exporto como getCryptosQuery para poder importarlo en los componentes al principio de cada uno.
Hago diferentes tipos de consultas para que por ejemplo me devuelva un cierto límite de criptomonedas, que me devuelva la información de cada criptomoneda específicamente y que obtenga también datos para los historiales gráficos. Al final también exportare diferentes palabras clave para poder hacer consultas en otros componentes.

Creo un archivo Store que es de los archivos más importantes ya que es un estado central de la verdad, es un sitio al cual se tiene acceso desde cualquier parte de mi aplicación pudiendo leer y modificar su contenido.

```
import { configureStore } from '@reduxjs/toolkit';
import { cryptoApi } from '../services/cryptoApi';
{/* Desde donde se obtienen los datos de Redux y la Api */}
export default configureStore({
reducer: {
[cryptoApi.reducerPath]: cryptoApi.reducer,
},
});
```

En este caso lo utilizare para los datos de la Api y también utilizaré la función de reducer que esta tomara el estado anterior de la Api y devolverá un nuevo estado, se utiliza para pasar valores por el array.

Lo siguiente será seguir desarrollando mi archivo criptocurrencies , aquí se especificará de qué manera quiero que se obtengan esos detalles de las criptomonedas y que quiero que se muestre una vez lo importe a mi App.js.

Creo una función con UseEffect que es usada para la inicialización de variables (en este caso cryptosList y searchterm) o llamadas a APIS.

Lo siguiente será buscar nuestra criptomoneda en la matriz de 100 criptomonedas distintas si esta es correcta o se encuentra esto nos devolverá el loader.

También creo el panel de búsqueda mencionado anteriormente en el diseño de la aplicación y con lo escrito en el código lo devuelvo en minúscula.

```
{/* creo funcion para que se ejecute cada vez que cambia cualquiera de estos dos valores: cryptosList, searchTerm */}
  useEffect(() => {
    setCryptos(cryptosList?.data?.coins);
{/* buscar la moneda en la matriz de 100 criptos */}
    const filteredData = cryptosList?.data?.coins.filter((item) => item.name.toLowerCase().includes(searchTerm));

    setCryptos(filteredData);
  }, [cryptosList, searchTerm]);

  if (isFetching) return <Loader />;

  return (
    <>
      {/* creo panel de busqueda */}
      {!simplified && (
        <div className="search-crypto">
          <Input
            placeholder="Search Cryptocurrency"
            onChange={(e) => setSearchTerm(e.target.value.toLowerCase())}
          />
        </div>
      )}
```

Aquí a continuación también creo diferentes containers para poder enlazarlo con mi CSS , crear diferentes links y tarjetas dinámicas para que se nos muestren las criptomonedas de una manera más presentable como mostrado anteriormente en el diseño de la página web, y en esta parte quiero que se muestran los siguientes detalles: Precio, Market cap (capitalización de mercado) y cuánto han cambiado en las últimas 24 horas.

El siguiente paso será crear mis detalles que mostraré en las criptomonedas asique tendré que crear otro componente llamado Crypto Details. Esta será la página secundaria de mi página web, donde tendremos muchos más datos disponibles de cada criptomoneda como mencioné anteriormente.

Para ello usare diferentes componentes de Ant design y de Millify y diferentes componentes de HTML para crear la estructura principal de la página secundaria.

También exportaré diferentes componentes que crearé en el paso siguiente como los gráficos que mostraré en esta página una vez entren en una criptomoneda.

```
const CryptoDetails = () => {
  const { coinId } = useParams();
  const [timeperiod, setTimeperiod] = useState('7d');
  const { data, isFetching } = useGetCryptoDetailsQuery(coinId);
  const { data: coinHistory } = useGetCryptoHistoryQuery({ coinId, timeperiod });
  const cryptoDetails = data?.data?.coin;
  { /* creo un cargador para devolver los datos de Linechart */ }
  if (isFetching) return <Loader />;
{ /* creo una matriz de tiempo */ }
  const time = ['3h', '24h', '7d', '30d', '1y', '3m', '3y', '5y'];
  { /* obtengo las estadisiticas */ }
  const stats = [
    { title: 'Price to USD', value: `$ ${cryptoDetails?.price && millify(cryptoDetails?.price)}`, icon: <DollarCircleOutlined /> },
    { title: 'Rank', value: cryptoDetails?.rank, icon: <NumberOutlined /> },
    { title: '24h Volume', value: `$ $ ${cryptoDetails?.["24hVolume"] && millify(cryptoDetails?.["24hVolume"])},`, icon: <ThunderboltOutline
    { title: 'Market Cap', value: `$ ${cryptoDetails?.marketCap && millify(cryptoDetails?.marketCap)}`, icon: <DollarCircleOutlined /> },
    { title: 'All-time-high(daily avg.)', value: `$ ${cryptoDetails?.allTimeHigh?.price && millify(cryptoDetails?.allTimeHigh?.price)}`, ico
  ];
  { /* obtengo las estadisticas genericas */ }
  const genericStats = [
    { title: 'Number Of Markets', value: cryptoDetails?.numberOfMarkets, icon: <FundOutlined /> },
    { title: 'Number Of Exchanges', value: cryptoDetails?.numberOfExchanges, icon: <MoneyCollectOutlined /> },
    { title: 'Aprroved Supply', value: cryptoDetails?.supply?.confirmed ? <CheckOutlined /> : <StopOutlined />, icon: <ExclamationCircleOutl
```

Como se muestra la siguiente imagen esta será la manera de la que haré los distintos tipos de consultas para luego mostrar todos los detalles de las criptomonedas mencionadas en el diseño de la página.

Así me permite hacer un const para luego poder usarlo para elegir diferentes temporalidades dentro de un gráfico, las estadísticas que se mostrarán primero y diferentes componentes que iré relacionando según avance con el desarrollo de la página web.

```
return (
  <Col className="coin-detail-container">
    <Col className="coin-heading-container">
      <Title level={2} className="coin-name">
        {data?.data?.coin.name} ({data?.data?.coin.symbol}) Price
      </Title>
      <p>{cryptoDetails.name} live price in US Dollar (USD). View value statistics, market cap and supply.</p>
    </Col>
    { /* creo un campo para escoger diferentes periodos de tiempo */ }
    <Select defaultValue="7d" className="select-timeperiod" placeholder="Select Timeperiod" onChange={(value) => setTimeperiod(value)}>
    { /* para recorrer nuestras opciones y que en cada opcion de muestre una fecha */ }
      {time.map((date) => <Option key={date}>{date}</Option>)}
    </Select>
    { /*  obtengo datos de Line Chart y creo un grafico */ }
    <LineChart coinHistory={coinHistory} currentPrice={millify(cryptoDetails?.price)} coinName={cryptoDetails?.name} />
    <Col className="stats-container">
      <Col className="coin-value-statistics">
        <Col className="coin-value-statistics-heading">
          <Title level={3} className="coin-details-heading">{cryptoDetails.name} Value Statistics</Title>
          <p>An overview showing the statistics of {cryptoDetails.name}, such as the base and quote currency, the rank, and trading volume.
        </Col>
```

Lo siguiente a lo que procedo es hacer un return . Creo distintas columnas donde pongo la descripción principal de lo que se mostrará a continuación.
También desarrolló distintos containers de CSS donde luego los enlazarse con los parámetros necesarios para que se muestre adecuadamente.
También creo el campo deseado para seleccionar las diferentes temporalidades. Lo siguiente será crear las siguientes columnas donde queremos mostrar la información y creo también diferentes containers para CSS, el primer párrafo de información también irá incluido aquí.

```
63          { /* obtengo datos de la API */ }
64          {stats.map(({ icon, title, value }) => (
65            <Col className="coin-stats">
66              <Col className="coin-stats-name">
67                <Text>{icon}</Text>
68                <Text>{title}</Text>
69              </Col>
70              <Text className="stats">{value}</Text>
71            </Col>
72          ))}
73        </Col>
74        <Col className="other-stats-info">
75          <Col className="coin-value-statistics-heading">
76            <Title level={3} className="coin-details-heading">Other Stats Info</Title>
77            <p>An overview showing the statistics of {cryptoDetails.name}, such as the
78          </Col>
79          { /* obtengo datos de la Api para icono valor etc */ }
80          {genericStats.map(({ icon, title, value }) => (
81            <Col className="coin-stats">
82              <Col className="coin-stats-name">
83                <Text>{icon}</Text>
84                <Text>{title}</Text>
85              </Col>
86              <Text className="stats">{value}</Text>
```

En la imagen que muestro a continuación muestra la manera que tengo de obtener los detalles de la Api con los diferentes parámetros creados, también muestro el orden en el que quiero que se muestran.
Y lo siguiente sería crear los parámetros necesarios para mostrar aún más información como historia detrás de la criptomoneda y diferentes links a sus páginas web y redes sociales creando distintos links que se muestran al final.

El penúltimo paso será crear nuestro gráfico donde se muestra el precio real de las criptomonedas. Para ello voy a crear dos arrays y que uno sea el precio de la cripto y otro el tiempo de la cripto,también tengo que crear diferentes bucles para que lean los datos del array de una manera determinada, en mi caso lo hago para que recorran cada criptomoneda una por una y por último creo otro bucle para que se lea el array del precio sobre el tiempo.

Procedo a crear distintas matrices una de ella para que me recoja el historial de precio y los datos, también les incluyo distintas etiquetas para enlazarlas con el historial como se muestra en la siguiente imagen englobando todo lo dicho anteriormente.

```
6   { /* creo el grafico que llame a distintas variables */ }
7   const LineChart = ({ coinHistory, currentPrice, coinName }) => {
8     { /* matriz para recorrer historial de criptos y obtener precios */ }
9     const coinPrice = [];
10    { /* matriz para recorrer historial de criptos y obtener marcas de tiempo*/ }
11    const coinTimestamp = [];
12    { /* creo diferentes bucles paraque salgan de 1 en 1 en la matriz de precio */ }
13    for (let i = 0; i < coinHistory?.data?.history?.length; i += 1) {
14      coinPrice.push(coinHistory?.data?.history[i].price);
15    }
16    { /* creo diferentes bucles para que salgan las marcas de tiempo de la matriz anterior */ }
17    for (let i = 0; i < coinHistory?.data?.history?.length; i += 1) {
18      const date = new Date(coinHistory?.data?.history[i].timestamp);
19      coinTimestamp.push(new Date(coinHistory?.data?.history[i].timestamp*1000).toLocaleDateString());  }
20      { /* que datos y opciones y lo relaciono con el array Timestamp */ }
21      const data = {
22      labels: coinTimestamp,
23      datasets: [
24        {
25          label: 'Price In USD',
26          data: coinPrice,
27          fill: false,
28          backgroundColor: '#0071bd',
29          borderColor: '#0071bd',
```

Creo un objeto de opciones para definir las escalas de la gráfica,y confijo los distintos ejes que utilizará la gráfica. Con esto habré terminado mi componente (gráfica).

Y ya por último una vez acabados todos los componentes que voy a usar para mi página web , me quedaría adaptar algunos componentes que explicaré a continuación para que mi pagina se muestre de cierta manera si se están utilizando dispositivos móviles.

Lo que quiero es que el menú sea desplegable y esté mostrado de otra manera en caso de que se esté visualizando en un dispositivo móvil, esto lo conseguiré importando use effect y use State, que son diferentes estados que nos permite usar React Native.

Con use effect creó una matriz de dependencia que se ejecuta al principio una sola vez y será para establecer el tamaño de la pantalla y con el segundo use effect lo que estoy haciendo es crear una condición dentro de él para que este se compruebe al ancho de la pantalla y en caso de que sea en un dispositivo lo detecta y devuelva el menú deseado.

Por último creo un botón ,que será para el menú con lo que hemos exportado antes.

Todo esto lo meteré dentro de mi componente NavBar que es donde tengo hecha toda la lógica del menú.

Con todo esto ya tengo mi página web lista para funcionar, por último ejecutar el comando: -npm start , para ver que se muestre todo adecuadamente.

9. Problemas encontrados

He tenido algunos problemas a lo largo del desarrollo del proyecto que no he encontrado solución. El primero es que la API de la que obtenía los datos al principio del proyecto podía obtener más información como noticias etc. Pero mientras seguía con el proyecto, sin previo aviso sacaron una versión de pago de la API la cual solo te dejaba obtener datos básicos como el precio de criptomonedas y poco más por lo que me vi obligado a tomar la decisión de acortar algunos elementos que iba a incluir en el menú .

El segundo problema que he tenido ha sido ahora al finalizar la aplicación, queria subirla a un dominio gratuito para poder enseñarla más adecuadamente, pero cuando ejecute el último comando que hay que ejecutar en una aplicación de react (npm build) , me aparecieron varios errores de sintaxis de espacios que hay en el código los cuales no he conseguido resolver a fecha de hoy, por lo que me veo obligado a subirlo más tarde a un dominio.

10. Evolución y trabajo futuro

Viendo el resultado de como ha quedado la página web no tengo duda alguna de que quiero seguir desarrollando y trabajando en ella. Como he mencionado anteriormente me gustaría crear más apartados dentro del menu:
-Noticias: noticias de última hora relacionadas con este ecosistema.
-Exchanges: una página dedicada a la cantidad de plataformas que existen e intercambio de criptomonedas para mostrar una información verificada y que aporte confianza para no caer en bulos.
-Proyectos dentro de cada ecosistema : en este apartado se mostrará los proyectos más grandes y la cantidad de proyectos que habría trabajando los ecosistemas de estos mismos.
-TVL: el TVL o Total Value Locked es la cantidad de liquidez que tienen bloqueada los inversores dentro de un proyecto para apoyar y asegurar el funcionamiento de este mismo, en este apartado se podrían crear gráficas también para mostrar la confianza que tienen los inversores en el proyecto.

11. Bibliografía

1.React | Qué es, para qué sirve y cómo funciona | Descúbrelo todo. (2021, 8 noviembre). Tribalyte Technologies. https://tech.tribalyte.eu/blog-que-es-react

2.Redux |Board, G. O. (2019, 24 diciembre). *¿Qué es Redux y cómo funciona?* Get on Board. https://www.getonbrd.com/blog/que-es-redux-y-como-funciona

3.Node JS |Simões, C. (2021, 27 julio). *¿Qué es Node.js y para qué sirve?* ITDO Desarrollo web y APPs Barcelona. https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/

3. Creador de Diagrama de Gantt Online. (2022). Venngage. https://es.venngage.com/features/hacer-diagrama-de-gantt